**Medium Scrapper**

**GitHub Link – ‘**[**https://github.com/pratyush-954/Pratyush\_Priyansh\_ECE\_Medium\_Scraper.git**](https://github.com/pratyush-954/Pratyush_Priyansh_ECE_Medium_Scraper.git)**’**

-By Pratyush Priyansh

ECE, 4th year

1729031038

ABES Institute of Technology, Ghaziabad

**Language Used-**

Python (v 3.6.0)

**Editor of Choice-**

PyCharm (Community Edition)

**Introduction-**

* This is a scraper written in Python.
* It can scrap certain data from [Medium.com](http://www.Medium.com).
* There are certain other functionalities to it.

**Modules used-**

* BeautifulSoup4
* Requests
* Pandas

**Contents-**

**S. No Topic Pg.No.**

1.) Introduction 1

2.) Modules Used 2

3.) Contents 2

4.) Problem Statement 3

5.) Methodology 4

6.) Working-

(i.) Part 1- The Scraping 5-6

(ii.) Part 2- Acquiring the content 7-9

(iii.) Part 3- Displaying the content 9-10

(iv.) Part 4 10

**Problem Statement-**

We need to scrape [Medium.com](http://www.Medium.com), in order to get the table of a few desired results, we want to have and display them in tabular form.

Apart from the display, we also need to store all those searched queries, as well as the results we obtain from the particular search query. That is, we need to form a database, which can be accessed later on by logging in to the admin portal.

**Methodology-**

* So, the idea is to first knock the medium search URL and receive a response from the server.
* Now, we collect the whole HTML code of that page in a proper manner and store it in a variable.
* Once, we have this code by our side, we can apply various actions on it, like finding, searching some “tag” in the code.
* So, in order to extract an intended data, we identify the similar tags which are particular to those data only.
* Once, have all these data with ourselves, we can store them in form of array, list or any other form we want.
* Now, as to display these results from the database meanwhile storing them, we need to extend a bit more.
* What we need is to link this above concept and logic to actual display intended results in real time.
* For, that we acquire a list and iterate within this list, displaying every element.
* As for the storing of the data, we first need to create an admin panel.
* All these data stored in database can be accessed by logging in to the admin portal.

**Part-1- The Scraping:** Day:1

**Libraries used**: Beautifulsoup4, Requests, Pandas, csv(optional), sqlite3(optional).

* To scrap the data from any website, we first need to acquire the structure or the HTML code of that website.
* This can be done by sending a request using the URL of that website and getting all of the content.
* Now, we can put and refine this content, using BeautifulSoup, where we can parse this content and store it in a proper variable for further processes.
* So, what we do have our content, we apply some logic for scraping.

**Logic Applied:**

* Now that page content is parsed, we can apply some actions to it, in order to achieve our goal.
* So, suppose we first need to get Titles of all the blogs, so first we will identify the common “tag”, like ‘h3’for example, that is present in every title.
* We can find all these tags and extract the title, as text format.
* Now this text with we have can be stored as a form of list.
* Repeating a similar process for other data which we need can, we can obtain all the lists.
* Now, we can parse all these obtain list to a main list.

**For creating and storing a data in csv file-**

* So, we can create a file with ‘.csv’, extension format.
* Here we can write all the data in this file and display into excel sheet.
* But this file content will be over written each time we make a new query, unless we change the name of file in the code itself.
* Although, this file can be used for a proper go.

**For creating and storing a data in local database (BD file)-**

* For this we use sqlite3 database.
* So, we first need to connect to database, where we name our database now.
* Next, we create we fields for the content of the table to be stored in the database.
* Next, we fill on these contents, by iterating through the main list.

**Part-2- Acquiring the contents:** Day:2

**Libraries used:** render, requests.

**Framework:** Django**,** Bootstrap

* To start off with this we first need to make a back end profit, where all these logics will work.
* We first need several types of files, where we can do our work and then connect it all together.
* So, we first an app file. This app file contains all the necessary files like:
  + Manage.py- Perhaps the most import of all. This file is needed to make any command run in terminal for our goal, be it creating a database or running our server, every time we need this file.
  + url.py- Here we need to create our path, i.e., every time we make a new response, where actually our page will take us next.
  + Views.py- In this we have all our necessary libraries and all the functions which has logics and all the processes are kept here.
  + Index.py- Now, this file can be said as front end of our project as here, we put all the necessary code of frontend display like buttons, search bars, title bar etc. This is also where all the display format is kept.
  + Models.py-A very crucial file for our purpose. Here, we can define the class format of all the collect data and provide them with their proper orientation, by simply importing this file to our views.py file.
  + Setting.py- This file has defaulted all the necessary data that we need. We can make a few changes according to our need in installed apps or change the time zone etc.
  + Admin.py- In this file we can register our model class so, that they can be stored in the database and we can also access them by logging in.

Now, we shall see how everything is actually happening.

* First of all, we need to make a local server for our site access.
* We can create a local server using Django framework.
* By default, we have 127.0.0.1:8000; here 8000 is our port number and we also change it.
* But we now need our own homepage, and we need something to display on the homepage.
* So, for that we create some search bar or navigation tool or buttons using **Bootstrap**, in our index.html file.
* Now, we need to route our path so, that whatever frontend we created can be displayed.
* We can create this path in url.py file.

Now, suppose we want to enter a query so that as input and we want to actually search this query.

* For that we have to make a function which can have this query as data input and also, here we can inherit the function for scraping of the data.
* Now, we need to set a path that were we will actually be routed once our query is entered.
* Now, we also need to have some input medium for it. So, we can create a button in index.py and this button will have the path to route to the URL containing the function of the input data and scraping.

**Part-3- Displaying the contents:** Day:2

* Now, we need to display our content.
* For that we take all our data and put that data in form of list of dictionaries.
* We make this dictionary so that every content is below its desirable format.
* Now, once we have the data, we make a class model of our data.
* We now import this model in our view function and make an object of this class.
* After that we iterate all the desirable data and make them under desirable format.
* Now, as for the tabular display we can do that in index.html file.
* We also need to migrate our data in order to make it accessible to the table.
* We are now ready to display our content.
* In order to save it to our database we need to make it accessible and register our model in the admin.
* Once, registered we can now create super user to log in to the admin portal, via providing proper credentials.

**Part-4- Deployment:** Day:3

* Done some beautification on the front end.
* Fixing some known errors.
* Hosting on the free service provided by Heroku.

I would like to thank you for providing me with this opportunity.

I learnt a lot via this project and get to know many aspects of programming.

Thank You.